Chapter 8

How to Use CVS?

8.1 What is CVS?

CVS is an acronym for the "Concurrent Versions System". It is a "Source Control" or "Revision Control" tool having the following features:

- Non-proprietory and can be downloaded from the internet;
- Allows users to work simultaneously on the same file, keep track of changes by revision, tag and date;
- Can obtain an ealier version of the software easily;
- Allows the user to track the supplier's software releases while making code changes locally.
- Enables the user to merge code changes between his version and supplier's automatically and identify problems if merge presents contradictions;
- A user of CVS needs only to know a few basic commands to use the tool.

Here are some important terms used with CVS:

Repository: The directory storing the master copies of the files. The main or master repository is a tree of directories.

Module: A specific directory (or mini-tree of directories) in the main repository. Modules are defined in the CVS modules file.

RCS: Revision Control System. A lower-level set of utilities on which CVS is layered.

Check out: To make a copy of a file from its repository that can be worked on or examined.

Revision: A numerical or alpha-numerical tag identifying the version of a file.

8.2 How to Use CVS?

There are two ways you can use CVS:

1. Use CVS to keep up to date with the GMI code changes. This will require a *sourcemotel* account.
2. Use CVS to track both GMI code releases and your own changes. Again you can do this either on *sourcemotel* or on your local machine (with your own CVS installation).

8.3 Use CVS to Keep Up to Date with GMI Source Code Changes

CVS is used to keep track of collections of files in a shared directory called "The Repository". Each collection of files can be given a "module" name, which is used to "checkout" that collection. After checkout, files can be modified (using your favorite editor), "committed" back into the Repository and compared against earlier revisions. Collections of files can be "tagged" with a symbolic name for later retrieval. You can add new files, remove files you no longer want, ask for information about sets of files in three different ways, produce patch "diffs" from a base revision and merge the committed changes of other developers into your working files. In this section, we explain how these operations are done with the GMI code. It is assumed that the user has an account on *sourcemotel* and that CVS is installed on his local computer.

We assume that you have already obtained a copy of the code (say the most recent release) from *sourcemotel* by using the command:

```
%cvs -d usrid@sourcemotel.gsfc.nasa.gov:/cvsroot/gmi co gmi_gsfc
```

A directory labeled *gmi_gsfc* (containing the code) will be created at the location where the command was executed.

Assume that you want to know all the different available releases (with the associated tags) of the GMI code. From the *gmi_gsfc* directory, type

```
%cvs status -v Makefile
```

to obtain the status of the file *Makefile*. The results give (first few lines):

```
===================================================================
File: Makefile          Status: Up-to-date

   Working revision:    1.18
   Repository revision: 1.18    /cvsroot/gmi/gmi_gsfc/Makefile,v
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      (none)

   Existing Tags:
        fromGEMHOME_to_GMIHOME          (revision: 1.18)
        NOxEmissionsScalingFactors_v3   (revision: 1.17)
        NOxEmissionsScalingFactors_v2   (revision: 1.17)
        Lightning_Branch_fvgcm_ap1_0    (revision: 1.14)
        NOxEmissionsScalingFactors_v1   (revision: 1.17)
        Lightning_Branch_fvgcm_default  (revision: 1.14)
        Lightning_Branch_DAS_default_fix        (revision: 1.14)
        Lightning_Branch_DAS_default    (revision: 1.14)
        Lightning_Branch_DAS_ap1_0      (revision: 1.14)
        SurfaceConstituents_for_ColumnDiagnostics       (revision: 1.17)
        ImplementationMEGANv1           (revision: 1.17)
        Flux_Freq_Routines_in_Modules   (revision: 1.16)
        GeorgiaTechCloudModule_v2       (revision: 1.16)
        HorizontalDomainForFreqOutputs  (revision: 1.16)
        GeorgiaTechCloudModule_v1       (revision: 1.16)
```

```
%cvs export [-D today][-r tag] gmi_gsfc
```

gives exported version of the *gmi_gsfc* directory. The expressions in [ ] are options. '-D today' gives the latest version of the code. The user can also specify "-D 'September 26, 2007'" (note that the date is in single quote) for version of that day, or use '-r release-1-17' for release 1.17 (release-1-17 is a CVS tag), or '-r NOxEmissionsScalingFactors_v3' for the *gmi_gsfc* directory with tag NOxEmissionsScalingFactors_v3.

```
%cvs checkout gmi_gsfc
```

provides in addition to exported version, CVS information. With such information, users will be able to keep up-to-date with our release automatically with simple `cvs update` command (instead of having to manually insert the changes we broadcast). Once you check out a version of the code, you form a 'working directory'.

```
%cvs update gmi_gsfc
```

only works if a user has cvs-checked-out version. This brings the changes made in the master repository to the user's working directory. An example of the print out from this command:

Example 1 *You want to checkout a copy of the GMI code from* sourcemotel*. Then type*

```
%cvs checkout gmi_gsfc
```

It creates a copy of the code in your own directory. Assume you have made some changes in the code and the next code release arrives. You can simply do a `cvs update` *to bring the new changes in the new release into your copy:*

```
%cd gmi_gsfc
%cvs update
```

A list is printed on your screen to let you know which files were updated (a 'U' in front of the file) from the new release, and which files were modified (a 'M' in front of the file) and any conflict that may result from this update.

Remark 7 *Note that doing* **cvs update** *under the* gmi_gsfc *directory will automatically update the entire code. You can update individual directory or file by going into the directory and do* **cvs update**- *which updates that directory and any sub-directories, or* **cvs update filename**- *which updates only that file.*

```
%cvs diff filename
```

This does the differencing between the file in your working repository with the one you checkout from the *sourcemotel* repository.

Example 2 *Assume that you want to compare the file* Makefile *(inside gmi_gsfc) from your working repository with the one on* sourcemotel *in the release with TAG* NOxEmissionsScalingFactors_v3*:*

```
%cvs diff -r NOxEmissionsScalingFactors_v3 Makefile

Index: Makefile
===================================================================
RCS file: /cvsroot/gmi/gmi_gsfc/Makefile,v
retrieving revision 1.17
retrieving revision 1.18
```

```
diff -r1.17 -r1.18
9,11c9
< LIBS = -L$(LIB_DIR)
<
< #FFLAGS+=$(INCS)
---
> LIBS = -L$(LIB_DIR)
38,39c36,37
< #all: packageddir shared components
< all: packageddir shared components applications
---
> new: packageddir shared components applications
> all: packageddir shared components applications legacy
48a47,49
> legacy:
>       (cd $(GMIHOME)/gem; $(MKMF); make)
>
57,58c58,59
< #     @$(MAKE) -C $(APPLICATIONS) EmissionDriver.ex
< #     @$(MAKE) -C $(APPLICATIONS) DiffusionDriver.ex
---
> #       @$(MAKE) -C $(APPLICATIONS) EmissionDriver.ex
> #       @$(MAKE) -C $(APPLICATIONS) DiffusionDriver.ex
68a70
>       (cd $(GMIHOME)/gem; make clean)

%cvs log filename
```

This lists the log messages and status of the master repository.

8.4 Use CVS to Track Both New Releases and Your Changes

If you want to maintain your own code and keep track of the changes from *sourcemotel*, what you should do is create your own repository and use the 'vendor branch' concept in CVS. If you do it from your local machine, set

```
setenv CVSROOT some-home-directory-on-your-local-machine
```

(e.g. setenv CVSROOT /home/userid/gmi_ repository)
in your *.cshrc* file.
To initialize the repository, type

```
%cd /home/userid/gmi_repository
%cvs init
```

Now you can checkout any release of the GMI code. Assume that you want to obtain the release *HorizontalDomainForFreqOutputs*

```
%cvs -d usrid@sourcemotel.gsfc.nasa.gov:/cvsroot/gmi co -r  \
     HorizontalDomainForFreqOutputs gmi_gsfc
```

If you only want the *Components/* directory, type

```
%cvs -d usrid@sourcemotel.gsfc.nasa.gov:/cvsroot/gmi co -d Components -r  \
     HorizontalDomainForFreqOutputs gmi_gsfc/Components
```

You can now work with the code. If you make some changes and want to bring them to your repository for keep, do the following from the directory *gmi_ repository/gmi_ gsfc*:

```
%cvs diff > output.diff
%cvs update
%cvs commit -m 'message for the commit'
```

If you want to create a new file that does not exist in the repository and you want to add it in the repository, type (from the directory where the new file resides)

```
%cvs add new_file
```

Example 3 *Assume that you want to add a new chemical mechanism,* new_chem*. In the directory* actm/gmimod/chem*, you have created the directory* new_chem/ *that contains the subdirectories* include_setkin/ *and* setkin/*. To add the directory structure of the new chemical mechanism into the repository, do the following:*

```
%cd Components/GmiChemistry/mechanisms
%cvs add new_chem/
%cvs commit new_chem/
%cd new_chem
%cvs add include_setkin/
%cvs commit include_setkin/
%cd include_setkin
%cvs add *
%cd ../
%cvs add setkin/
%cvs commit setkin/
%cd setkin
%cvs add *
```

8.5 Where To Obtain CVS?

```
https://ccvs.cvshome.org/servlets/ProjectDocumentList
```